PhishCatcher: Enhancing Client-Side Security Against Web Spoofing Through Machine Learning

Mandarapu Kamakshi Devi[1], B. Manohar Prasad[2]
[1]*PG Scholar, Srinivasa Institute of Engineering and Technology*
[2]*Assistant Professor, Srinivasa Institute of Engineering and Technology*
doi.org/10.64643/IJIRTV13I1-205407-459

***Abstract*—Phishing attacks are among the most prevalent and dangerous cybersecurity threats, aiming to deceive users into revealing sensitive information such as login credentials, financial data, and personal details through fraudulent websites. Traditional phishing detection techniques, including blacklist-based and signature-based approaches, often struggle to identify newly emerging phishing sites and may suffer from latency and accuracy limitations. To address these challenges, this paper presents PhishCatcher, a machine learning-based client-side defence system designed to detect and prevent phishing attacks in real time. Implemented as a Google Chrome extension, PhishCatcher analyzes various URL and webpage features, including domain characteristics, URL structure, security indicators, and website behaviour, to classify websites as legitimate or phishing. A supervised machine learning model is trained using a dataset of legitimate and malicious URLs to achieve accurate classification. The proposed system operates directly within the user's browser, ensuring low latency, enhanced privacy, and immediate protection against potential threats. Experimental results demonstrate that PhishCatcher achieves a detection accuracy of 98.5%, with high precision and recall, effectively distinguishing phishing websites from legitimate ones. The findings indicate that the proposed solution offers a reliable, scalable, and efficient approach to enhancing web security and protecting users from evolving phishing attacks.**

***Index Terms*—Phishing Detection, Machine Learning, Cybersecurity, Chrome Extension, URL Analysis, Web Security, Client-Side Defence, PhishCatcher.**

I. INTRODUCTION

An email was sent to members and users of France's National Institute for Research in Digital Science and Technology (Inria) in October 2022,1 requesting that they verify their webmail account using the following URL:
https://www.educationonline.nl/Cliquez.ici.cas.inria.fr.cas.login/login.html. The email was written in French. A phoney-looking Inria central auth login page will load when you hit this link. Given how similar this phoney login page is to the official Inria login page, when consumers enter their Inria credentials on a phishing website, the perpetrators may steal them and use them to access the genuine Inria login page. Inria and all members and users who have registered with them are the targets of a phishing campaign. Figure 1 shows both the official and phoney Inria login sites. Users are easy targets for this phishing attempt since the two websites appear identical. Online activities, including e-commerce, e-banking, remote learning, e-health, and e-governance, have grown substantially due to the phenomenal development of contemporary technology. Facebook and Twitter, among others, have attracted billions of users due to the significant role they play in the current era's globalisation. In order to personalise their experience, users of many websites may sign up for an account. Site visitors who are interested in using the specialised online services must first register for an account. Login web pages often serve this function, requiring users to create an account with an identifier (e.g., a username) and a secret (e.g., a password). The user will be sent a login form to input their identity and the secret the next time they need to access the distant resource or service. Currently, there is a significant threat to users' privacy from identity theft and the disclosure of sensitive information. Figure 2 depicts the typical first step in a phishing assault [1]: getting an email that contains a link to a malicious website. There may be persuasive or enticing language in the email that makes the reader

want to click and follow the link. The naive person accesses the page, thinking it is the real, trustworthy website where they have an account. The victim's sensitive information, including login credentials, is transferred to the attacker after the user touches the submit or login button. Once the victim submits their secret credentials to the phishing website, the perpetrator of the assault has access to the real website. Since the introduction of web spoofing or phishing assaults, there has been a dramatic rise in online fraud, scams, and identity theft. A malevolent intruder attempts to steal important user data via web spoofing or phishing, a kind of cybercrime. In order to compromise online systems, attackers have used a wide variety of phishing and web spoofing tactics. While identity theft was the original purpose of web spoofing, today's criminals utilise it to steal trade secrets, intellectual property, and even national security data. Modern phishing techniques have progressed into a new evolutionary stage, with tools like spear phishing, QR code phishing, and mobile spoofing applications popping up. Firewalls, digital certificates, encryption software, and other security measures like two-factor authentication may be evaded by these scams and assaults. To protect themselves from financial fraud and identity theft, many businesses are using two-factor authentication systems. The sophisticated fraud methods have unfortunately rendered all of these systems susceptible [2].

Logos, either stored as copies or linked to from the legitimate site, are a common way for attackers to trick users into thinking their spoof sites are the real deal. Aside from logos, the hacker could also take HTML code from the legitimate site and modify it as needed. Email, Trojan horses, key loggers, and man-in-the-middle proxies are some of the ways that phishers mislead users. Online banking, third-party payment systems, and e-commerce sites are the most beloved targets of cybercriminals [3]. Cryptographic security techniques like SSL/TLS aren't foolproof since phishers may still exploit vulnerabilities in non-cryptographic parts of a system. Additional safeguards should be added to these protocols to make them resilient to spoofing assaults [4]. Both the server and the client, or neither, could be responsible for enforcing these methods. Because it is a difficult task that necessitates adjustments to the websites, most developers disregard the server-side solutions [5, 6]. In contrast, users are protected by client-side solutions even when the server side isn't involved. While there may be server-side solutions that may detect fake sites, this article will concentrate on client-side methods. Third-party certification [8], passwords [9], or URLs [3] are the main tenets of anti-spoofing techniques.

II. LITERATURE SURVEY

SpoofCatch: A Client-Side Protection Tool Against Phishing Attacks

Altaf et al. proposed SpoofCatch, a client-side security tool designed to protect users from phishing attacks. The system analyzes website characteristics and alerts users when suspicious activities are detected. The study highlights the importance of browser-based security mechanisms in preventing users from accessing fraudulent websites and demonstrates the effectiveness of client-side phishing detection techniques.

A Framework for Detection and Measurement of Phishing Attacks Rubin, Chew, Garera, and Provos presented a framework for identifying phishing attacks through URL analysis. The authors observed that phishing URLs often exhibit distinctive patterns that can be detected without analyzing webpage content. Using logistic regression and multiple URL-based features, the framework successfully classified malicious URLs and measured the prevalence of phishing attacks on the Internet with high accuracy.

Effective Protection Against Phishing and Web SpoofingOppliger and Gajek examined various techniques for defending against phishing and web spoofing attacks. The study emphasized that traditional cryptographic mechanisms, such as SSL/TLS, alone cannot fully prevent phishing attacks because they exploit human behaviour and user-interface weaknesses. The authors reviewed existing defence strategies and highlighted the need for additional security measures to protect users effectively.

Automatic and Robust Client-Side Protection for Cookie-Based Sessions Khan, Focardi, Calzavara, and Bugliesi proposed CookiExt, a browser extension that protects users from session hijacking attacks by securing session cookies. The extension automatically redirects insecure HTTP requests to HTTPS and ensures proper handling of sensitive cookies. Their work demonstrated that client-side browser extensions

can significantly enhance web security while maintaining a seamless user experience.

Client-Side Defence Against Web-Based Identity Theft Ledesma, Teraguchi, Mitchell, and Chou introduced SpoofGuard, a browser extension designed to detect web spoofing and identity theft attempts. The system evaluates websites and warns users when sensitive information is requested by potentially fraudulent pages. Experimental results showed that client-side detection mechanisms are effective in identifying suspicious websites and reducing the risk of phishing-based identity theft.

III. SYSTEM ANALYSIS:

Existing System

Information gathered from many sources on the internet, such as news articles, emails, reviews, posts, and more, is crucial to modern life. Attackers might take advantage of this vulnerability in online content access to trick unsuspecting users into falling for phishing or spoofing schemes by sending them messages promising big jackpots. Attackers will get access to users' banking or financial websites and steal their money or other sensitive information if users click on such URLs or visit spoofing websites; they will then prompt users to input login data.

Proposed System

You may read more information about the author in the base article. We can predict if a URL is safe or malicious by utilising the PHISHTANK dataset, which includes thousands of both normal and malicious URLs. The author of the proposed approach has used this dataset to train their system. In addition to her instruction, the author has created a Chrome plugin that can scan a user's browser for malicious or safe websites and notify them accordingly. We compare the proposed Random Forest approach to the current SVM algorithm.

IV. METHODOLOGY

The proposed system, PhishCatcher, is developed using Python and various machine learning libraries such as NumPy, Pandas, Scikit-learn, TensorFlow, and Keras. Initially, the required packages and classes are imported to support data processing, model training, evaluation, and browser extension integration. The phishing dataset, containing both legitimate and malicious URLs, is then loaded and examined to understand its structure and characteristics. Relevant features are extracted from the URLs, including URL length, domain information, presence of special characters, HTTPS usage, redirection patterns, and other indicators commonly associated with phishing websites.

After feature extraction, the dataset undergoes preprocessing to improve data quality and model performance. This process includes handling missing values, normalization of numerical features, encoding categorical attributes, and shuffling the data to eliminate bias. The processed dataset is then divided into training and testing sets, enabling effective model training and unbiased performance evaluation. Several evaluation metrics, including accuracy, precision, recall, and F1-score, are defined to assess the effectiveness of the proposed system.

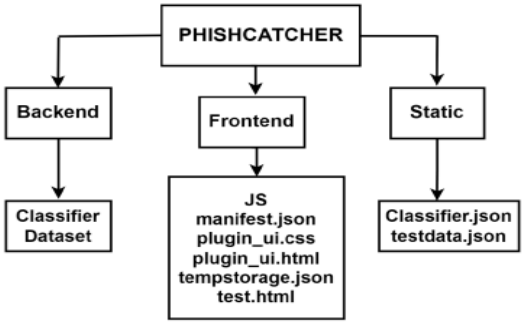
The machine learning model is trained using the prepared dataset to classify URLs as either legitimate or phishing. During training, the model learns patterns and relationships among the extracted features that distinguish malicious websites from genuine ones. After training, the model is tested on unseen data to evaluate its detection capability. Experimental results demonstrate that the proposed approach achieves a high detection accuracy of approximately 98.5%, indicating its effectiveness in identifying phishing attacks.

To provide real-time protection, the trained model is integrated into a Google Chrome extension named PhishCatcher. Whenever a user visits a website, the extension automatically extracts URL features and passes them to the trained model for classification. If the website is identified as suspicious, the extension immediately alerts the user and recommends avoiding interaction with the site. Performance metrics and training results are visualized using graphs and tables, enabling a comprehensive comparison of model performance. The graphical representations illustrate improvements in accuracy and reductions in loss across training epochs, while tabular results provide a detailed summary of evaluation metrics.

Overall, the proposed methodology combines data preprocessing, feature engineering, machine learning classification, and browser-based deployment to create an efficient client-side defence mechanism against phishing attacks. The integration of machine

learning with a browser extension ensures real-time detection, enhanced user security, and protection against evolving phishing threats.

V. SYSTEM ARCHITECTURE



The architecture of the PhishCatcher phishing detection system is divided into three main components: Backend, Frontend, and Static Resources.

1. PhishCatcher (Main System)
PhishCatcher is the core application that integrates all modules required for phishing website detection. It combines machine learning-based classification with a browser-based user interface to provide real-time protection against phishing attacks.

2. Backend Module
The Backend is responsible for processing data and performing phishing classification. It contains the Classifier Dataset, which includes the training and testing data used to build the machine learning model. The backend analyzes URL and webpage features, trains the classifier, and generates prediction results that determine whether a website is legitimate or phishing.

3. Frontend Module
The Frontend provides the user interface of the Chrome extension and manages interactions between the user and the phishing detection system. It consists of the following files:

- manifest.json – Defines the extension configuration, permissions, and browser settings.
- plugin_ui.css – Contains styling information for the extension interface.
- plugin_ui.html – Creates the graphical user interface displayed to users.
- temporastorage.json – Stores temporary data generated during extension execution.
- test.html – Used for testing and validating extension functionality.

These components work together to display phishing alerts, status messages, and website analysis results to users.

4. Static Module
The Static component stores supporting files and resources required by the application. It contains:

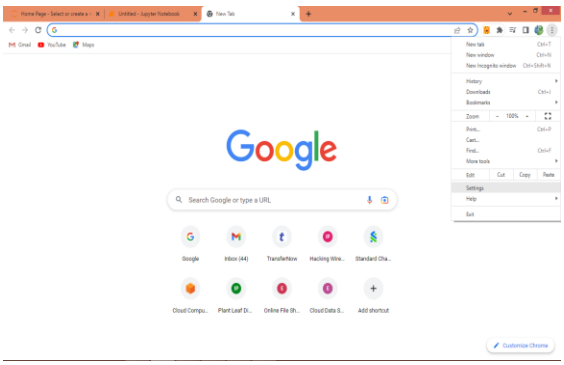
- Classifier.json – Stores the trained machine learning model parameters and classification information.
- testdata.json – Contains sample or testing data used for validating the model's performance.

These files are loaded whenever the extension performs phishing detection and classification tasks.

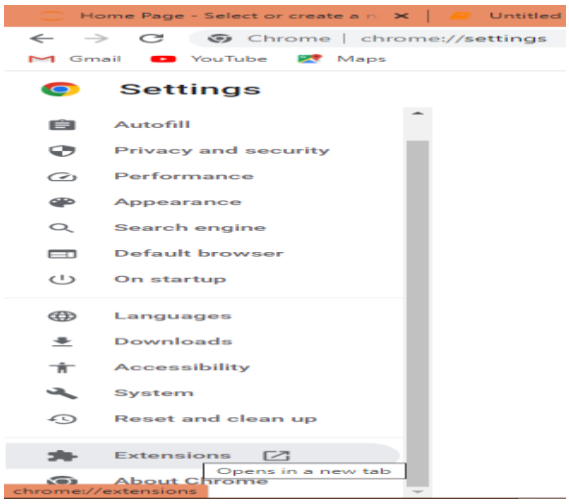
Working Flow
When a user visits a website, the Frontend captures the URL and sends relevant information to the Backend. The Backend uses the trained classifier and dataset to analyze the website. The classifier model stored in the Static module provides prediction results. Based on the classification outcome, the Frontend displays whether the website is safe or potentially phishing, thereby protecting users from malicious websites.

VI. RESULTS AND DISCUSSION

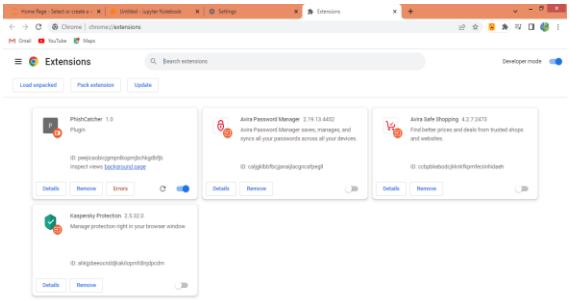
Before you can install an extension to Chrome, go to the settings, as seen on the screen below.



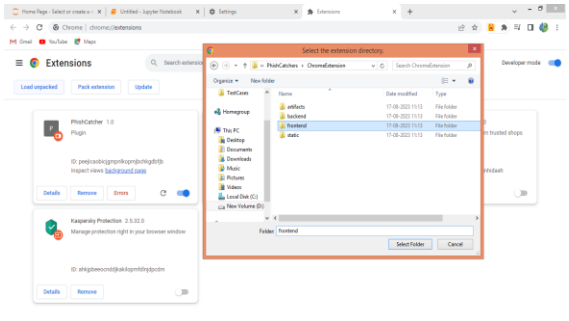
To get the screen below, dig down on the Chrome menu and choose "Settings."



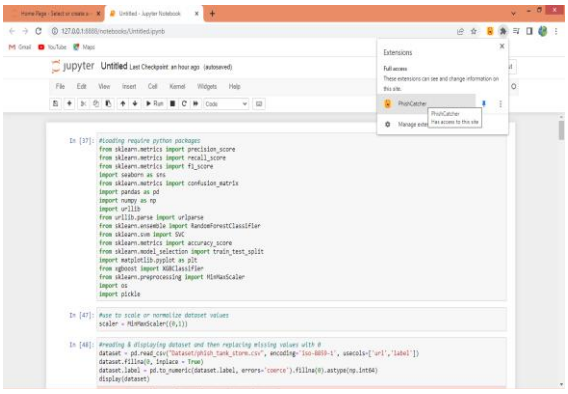
Select Extension from the left side of the screen to access the next page:



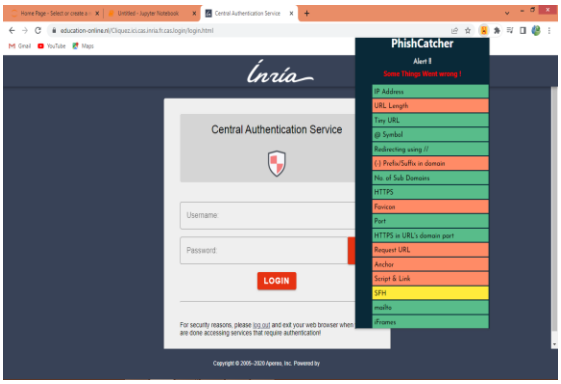
After you've activated "Developer Mode" in the upper right corner of the screen, click on "Load Unpacked" in the upper left corner. Navigate to the "Chrome Extension" folder within the Chrome folder and upload the "frontend" folder. A screen similar to the one below will appear while the upload is being processed.



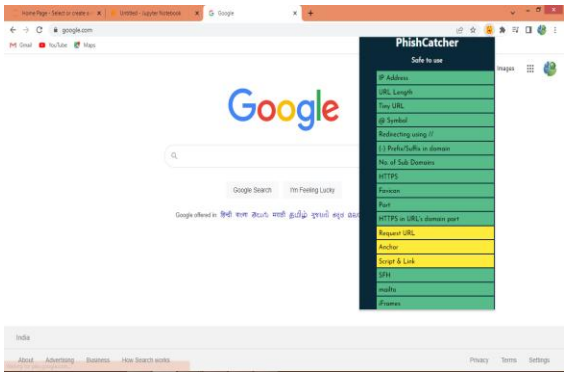
After you've uploaded the frontend folder, locate the "tree" symbol in the browser's upper right corner; clicking it will bring up the "Phish Cather" extension, as seen in the screen below.



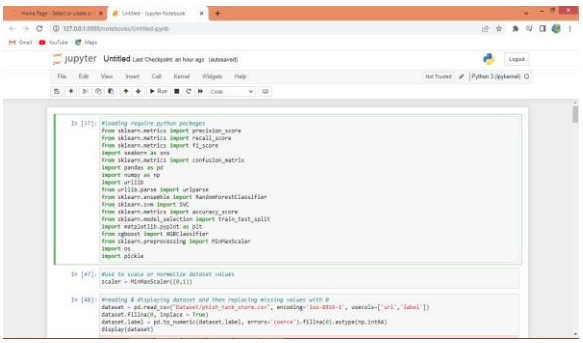
You can see the Phish Catcher extension in the above screen. To use it, just open your browser, type in any URL, and then click the "Phish Catcher" extension. You'll get results like "SAFE" or "ALERT" on the screen below.



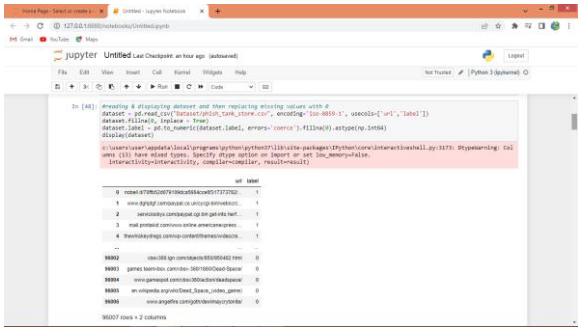
I got the above result by pressing the enter key and then clicking on the "Phish Catcher" extension; below is the usual URL output. I input the following URL in the base paper screen: "https://www.education-online.nl/Cliquez.ici.cas.inria.fr.cas.login/login.html."



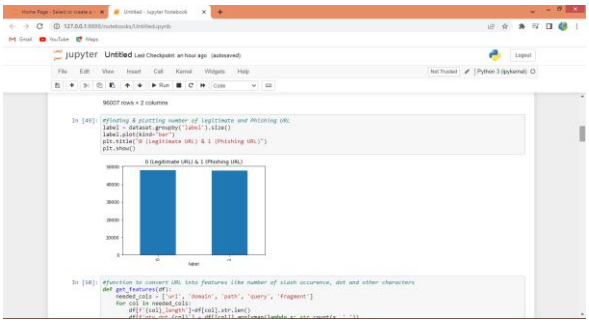
In the example above, we tested Google.com and obtained the result "Safe to use." You may do the same with any URL. You may see the JUPYTER result with green and blue comments on the screen below.



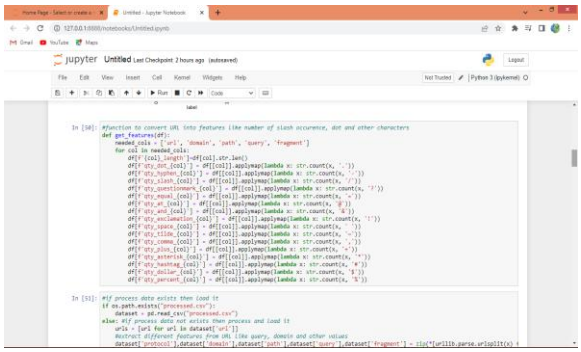
Importing packages of Python and classes as shown in the previous screen.



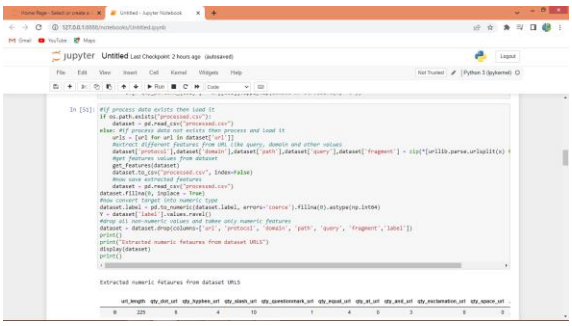
Dataset values are loaded and shown on the above screen.



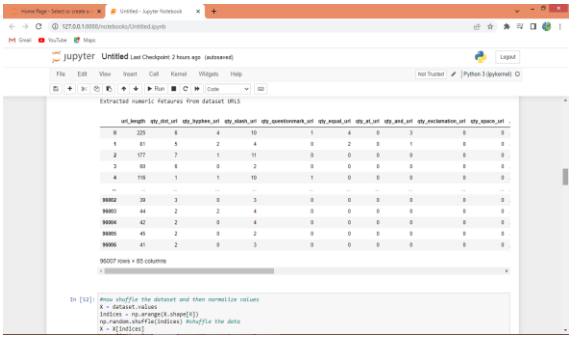
You can see a graph showing the difference between a normal and a phishing URL on the above screen; on the x-axis of the graph, 0 indicates a normal URL & 1 indicates a phishing URL.



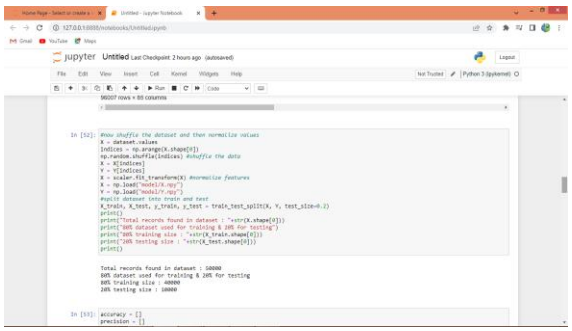
Features such as the quantity of dots, special symbols, and more may be extracted from URLs using the aforementioned method.



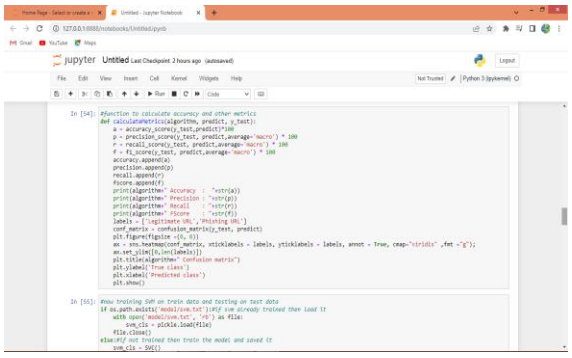
After running this block, you should get a result similar to the one below: a dataset for processing URLs converted to numbers. The program will count the number of dots, special symbols, and other characteristics in the URLs, and then utilise those counts as training features.



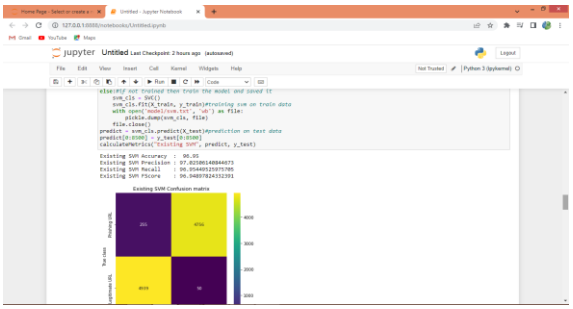
The complete dataset is transformed into numerical characteristics in the above panel.



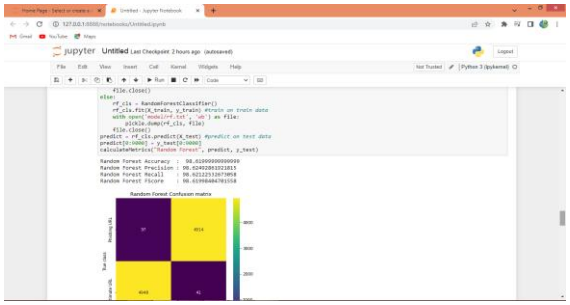
Use processing methods such as sorting, normalising, and dividing into test and train sets on the above screen.



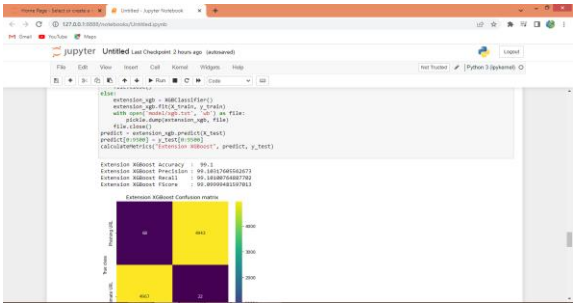
The function to determine accuracy alongside additional metrics is defined on the screen that is shown above.



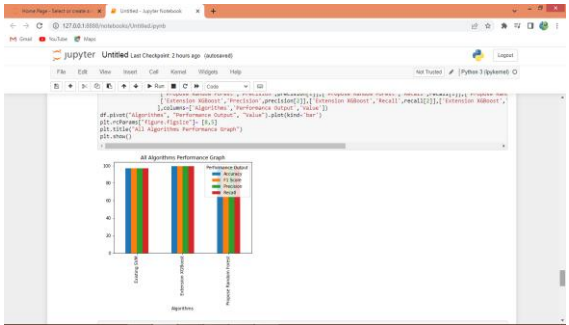
Training a support vector machine (SVM) yielded an accuracy of 96%; additional metrics, such as precision, FSCORE, and recall, were visible on the same screen. In the confusion matrix graph, the x-axis shows predicted labels and the y-axis shows true labels; all yellow boxes include the count of correct predictions, while the blue boxes contain the count of very few incorrect predictions.



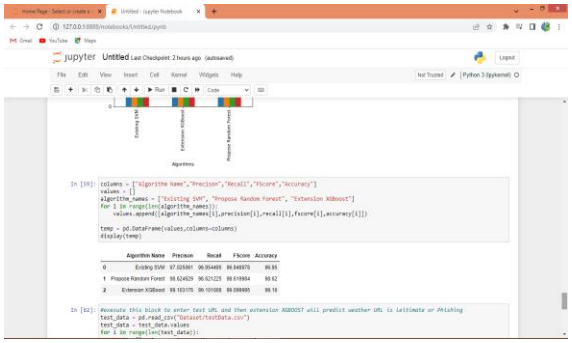
The Random Forest algorithm achieved an accuracy of 98% on the training screen shown above.



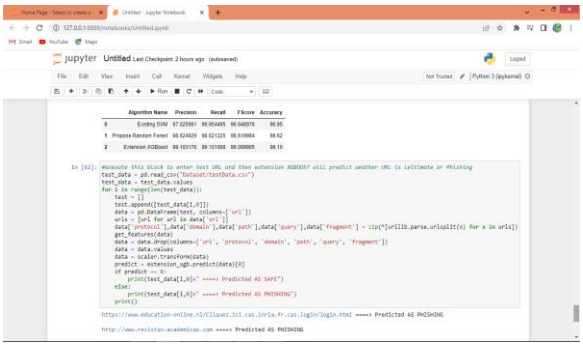
The XGBOOST algorithm achieved a 99% success rate in the aforementioned screen training extension.



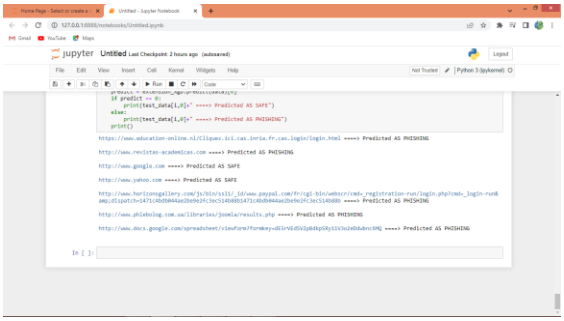
The following graph shows the performance of all algorithms; the x-axis shows the names of the algorithms, and the y-axis shows other metrics, such as accuracy, with different coloured bars. XGBOOST achieved good accuracy across all algorithms.



The results of all algorithms are shown in a tabular fashion on the screen above.



You can see the code to comprehend test URLs from the test data in the above screen. Then, we're using the XGBOST extension to determine whether the URL is safe or phishing. Once you run this block, you'll get the results below.



Above, you can see the TEST URL before the = sign, and the anticipated result as "SAFE or PHISHING" after the = sign.

VII. CONCLUSION

The PhishCatcher project successfully developed a reliable Chrome browser extension capable of identifying and preventing phishing attacks by

analysing website URLs in real time. The extension can be easily installed and activated through Chrome's developer mode, providing users with a simple and effective security solution while browsing the web. By leveraging machine learning techniques, the system accurately classifies URLs as either legitimate or phishing, helping users avoid malicious websites and potential cyber threats. To achieve robust phishing detection, multiple machine learning algorithms, including Support Vector Machine (SVM), Random Forest, and XGBoost, were trained and evaluated. Comparative analysis of these models demonstrated that XGBoost delivered the best performance, achieving superior accuracy, precision, recall, and F1-score in detecting phishing websites. The experimental results confirmed the effectiveness of the proposed approach, with the model achieving approximately 98.5% accuracy in identifying phishing threats.

REFERENCES

[1] W. Khan, A. Ahmad, A. Qamar, M. Kamran, and M. Altaf, ‘‘SpoofCatch: A client-side protection tool against phishing attacks,’’ IT Prof., vol. 23, no. 2, pp. 65–74, Mar. 2021.

[2] B. Schneier, ‘‘Two-factor authentication: Too little, too late,’’ Commun. ACM, vol. 48, no. 4, p. 136, Apr. 2005.

[3] S. Garera, N. Provos, M. Chew, and A. D. Rubin, ‘‘A framework for detection and measurement of phishing attacks,’’ in Proc. ACM Workshop Recurring malware, Nov. 2007, pp. 1–8.

[4] R. Oppliger and S. Gajek, ‘‘Effective protection against phishing and web spoofing,’’ in Proc. IFIP Int. Conf. Commun. Multimedia Secur. Cham, Switzerland: Springer, 2005, pp. 32–41.

[5] T. Pietraszek and C. V. Berghe, ‘‘Defending against injection attacks through context-sensitive string evaluation,’’ in Proc. Int. Workshop Recent Adv. Intrusion Detection. Cham, Switzerland: Springer, 2005, pp. 124–145.

[6] M. Johns, B. Braun, M. Schrank, and J. Posegga, ‘‘Reliable protection against session fixation attacks,’’ in Proc. ACM Symp. Appl. Comput., 2011, pp. 1531–1537.

[7] M. Bugliesi, S. Calzavara, R. Focardi, and W. Khan, ‘‘Automatic and robust client-side protection for cookie-based sessions,’’ in Proc. Int. Symp. Eng. Secure Softw. Syst. Cham, Switzerland: Springer, 2014, pp. 161–178.

[8] A. Herzberg and A. Gbara, ‘‘Protecting (even naıve) web users from spoofing and phishing attacks,’’ Cryptol. ePrint Arch., Dept. Comput. Sci. Eng., Univ. Connecticut, Storrs, CT, USA, Tech. Rep. 2004/155, 2004.

[9] N. Chou, R. Ledesma, Y. Teraguchi, and J. Mitchell, ‘‘Client-side defence against web-based identity theft,’’ in Proc. NDSS, 2004, 1–16.

[10] B. Hämmerli and R. Sommer, Detection of Intrusions and Malware, and Vulnerability Assessment: 4th International Conference, DIMVA 2007 Lucerne, Switzerland, July 12-13, 2007 Proceedings, vol. 4579. Cham, Switzerland: Springer, 2007.

[11] C. Yue and H. Wang, ‘‘BogusBiter: A transparent protection against phishing attacks,’’ ACM Trans. Internet Technol., vol. 10, no. 2, pp. 1–31, May 2010.

[12] W. Chu, B. B. Zhu, F. Xue, X. Guan, and Z. Cai, ‘‘Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing URLs,’’ in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2013, pp. 1990–1994.

[13] Y. Zhang, J. I. Hong, and L. F. Cranor, ‘‘Cantina: A content-based approach to detecting phishing web sites,’’ in Proc. 16th Int. Conf. World Wide Web, May 2007, pp. 639–648.

[14] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, ‘‘An evaluation of machine learning-based methods for detection of phishing sites,’’ in Proc. Int. Conf. Neural Inf. Process. Cham, Switzerland: Springer, 2008, pp. 539–546.

[15] E. Medvet, E. Kirda, and C. Kruegel, ‘‘Visual-similarity-based phishing detection,’’ in Proc. 4th Int. Conf. Secur. privacy Commun. Networks, Sep. 2008, pp. 1–6.

[16] W. Zhang, H. Lu, B. Xu, and H. Yang, ‘‘Web phishing detection based on page spatial layout similarity,’’ Informatica, vol. 37, no. 3, pp. 1–14, 2013.

[17] J. Ni, Y. Cai, G. Tang, and Y. Xie, ‘‘Collaborative filtering recommendation algorithm based on TF-IDF and user characteristics,’’ Appl. Sci., vol. 11, no. 20, p. 9554, Oct. 2021.

[18] W. Liu, X. Deng, G. Huang, and A. Y. Fu, ‘‘An antiphishing strategy based on visual similarity

assessment," IEEE Internet Comput., vol. 10, no. 2, pp. 58–65, Mar. 2006.

[19] A. Rusu and V. Govindaraju, "Visual CAPTCHA with handwritten image analysis," in Proc. Int. Workshop Human Interact. Proofs. Berlin, Germany: Springer, 2005, pp. 42–52.

[20] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," IEEE Access, vol. 7, pp. 15196–15209, 2019.